WWW.SCIAM-MAGAZINE.NAROD.RU

Занимательный компьютер

Моделирование поведения некоторых физических систем в домашней компьютерной лаборатории

А. К. ДЬЮДНИ

СОСУД, заполненный летающими во всех направлениях шариками и служивший ранее лишь воображаемой моделью физической среды, теперь можно наблюдать на экране домашнего компьютера. Приложив в общем-то совсем небольшие усилия к составлению программы, читатели смогут наблюдать за тем, как молекулярные столкновения создают давление в цилиндре или как в хаотическом движении красных и синих молекул происходит диффузионный процесс между двумя жидкостями. А те, кого не устрашает перспектива ядерного взрыва в их компьютере, могут даже поэкспериментировать с критическими массами нестабильного элемента, который я называю гридиумом.

Темой этой очередной статьи я обязан Дж. Блинну, научному сотруднику Лаборатории реактивного движения в Пасадине (шт. Калифорния). Блинн когда-то занимался разработкой компьютерных моделей различных физических систем. Его программы были записаны на довольно занимательной видеокассете с лаконичным названием «Механическая вселенная». Две модели Блинна привлекли мое внимание для описания в рубрике «Занимательный компьютер»: замечательные «мультипликации» цилиндра, наполненного газом, и камеры сгорания. В первом случае несколько десятков шариков прыгали, натыкаясь друг на друга и на стенки двумерного цилиндра, приводя подвешенный в цилиндре поршень в состояние «вибрирующего равновесия». Во второй модели шарики двух цветов совершают хаотическое движение в замкнутом сосуде. Первоначально на экране видны две области разного цвета, но затем по мере того как шарики совершают «броуновское» движение, цвета перемешиваются. Третья модель, которую мы рассмотрим в данной статье — ядерный взрыв, — это мое собственное изобретение. Его идея, наверное, объясняется безотчетным детским интересом к явлению взрыва.

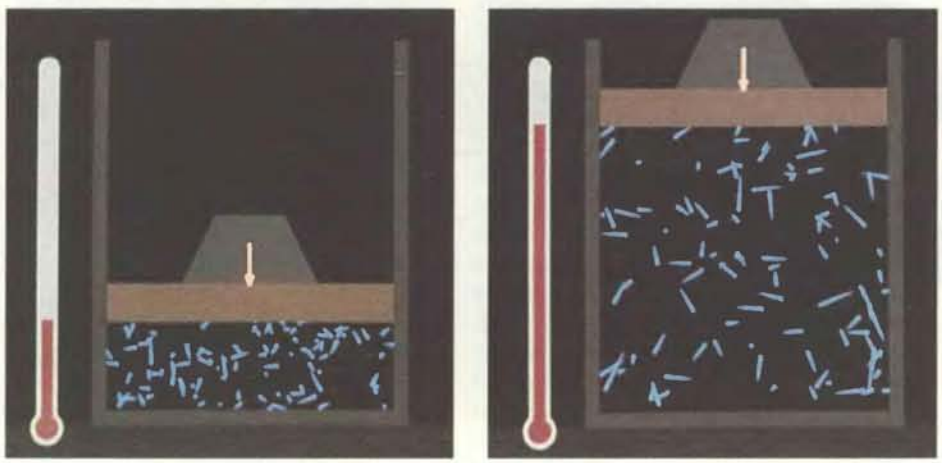
За те несколько часов, которые вам придется потратить на создание рабочей версии программы, названной мной BOUNCE (прыг-скок), вы будете с лихвой вознаграждены забавным зрелищем на экране. В модели Блинна шарики отскакивали друг от друга и от стенок сосуда, но можно написать значительно более простую и столь же эффектную программу, в которой шарики, игнорируя друг друга, отскакивают только от стенок цилиндра.

Пользователь программы BOUNCE может сам выбрать значения таких параметров, как скорость шариков и их количество. Положение, принимаемое поршнем, соответствует закону состояния идеального газа: $PV = kT$, т. е. произведение давления данной массы газа на занимаемый им объем пропорционально его температуре. Когда пользователь указывает скорость движения молекул, он тем самым выбирает определенное значение температуры, а вес подвижного поршня задает давление. Каким тогда будет объем? Ответ вы сможете ясно увидеть на экране (см. рисунок внизу). Чем выше температура газа, тем выше поднимается поршень.

Полезно проследить сначала за движением какого-нибудь одного шарика. Рассмотрим замкнутый сосуд, изображенный в виде прямоугольника на экране компьютера. Пусть это будет квадрат со стороной 50 условных единиц и пусть в какой-то точке, принадлежащей этому квадрату, находится наш шарик. Поскольку мы имеем дело с плоской моделью, положение шарика должно быть задано двумя координатными значениями, скажем x и y. Его движение характеризуется двумя координатами скорости v_x и v_y. За короткий промежуток времени t шарик переместится из положения (x, y) в положение $(x + t \cdot v_x, y + t \cdot v_y)$.

Программа BOUNCE рисует изображение цилиндра с поршнем также двумерное. У цилиндра три неподвижные стенки, определенные линиями $x = 0$, $x = 50$ и $y = 0$. Четвертую стенку представляет поршень. Ее уравнение $y = h$, где h — высота поршня в данный момент. Рано или поздно в программе, которая управляет системой движущихся шариков, придется рассмотреть вопрос, столкнется ли шарик с одной из стенок, и если столкнется, то когда. Программа BOUNCE сверяет направление движения шарика с каждым из четырех уравнений стенок цилиндра: в какой точке траектория движения шарика пересекает каждую из четырех линий? Ответы приведены в левой таблице нижнего рисунка на с. 88.

В каждом случае мы получаем значение времени t (отсчитанное от данного момента), через которое шарик столкнется с рассматриваемой стенкой. Координаты точки столкновения можно вычислить, подставив значение t в простые формулы, определяющие положение движущегося шарика. Если значение t получается отрицательным, это означает, что шарик, продолжая движение в том же направлении, никогда не натолкнется на



В модели Дж. Блинна объем газа увеличивается по мере повышения температуры

WWW.SCIAM-MAGAZINE.NAROD.RU

данную стенку, поскольку он движется в противоположном от нее направлении. Если значение t бесконечно велико, то одна из координатных составляющих скорости шарика должна быть равна 0. В этом случае шарик тоже никогда не столкнется с данной стенкой, и программа, конечно, не вычисляет значения t.

Первой стенкой, на которую натолкнется шарик, будет стенка с минимальным неотрицательным значением t. Полученным значением t можно воспользоваться, чтобы вычислить точку, в которой произойдет столкновение. Здесь нам пригодится небольшая табличка (см. правую половину нижнего рисунка на с. 88).

В большинстве программ, моделирующих поведение движущихся механических систем, используется один из двух возможных способов продвижения времени, или часового механизма модели. Если используется способ равномерного разбиения времени, то часы каждый раз продвигаются на короткий промежуток времени и положение всех компонентов системы равномерно изменяется. При использовании способа «критического события» часы продвигаются до того момента, когда происходит какое-то интересное явление. Читатели, наверное, помнят программу, моделирующую звездные скопления. Речь о ней шла в мартовском номере журнала за 1986 г. Там расположение всех звезд рассматриваемой системы изменялось с интервалом времени в несколько лет. В программе BOUNCE удобнее воспользоваться вторым приемом — ожиданием критического события, которое в данном случае происходит, когда шарик отскакивает при столкновении со стенкой.

При использовании подхода с ожиданием критического события очень важную роль играет массив *event* (событие). Если программа BOUNCE жонглирует n шариками, то она должна знать, какой из них следующим столкнется со стенкой. Поэтому массив на самом деле представляет собой очередь. Вот как действует программа:

repeat

1. получить значения *времени*, *шарика* и *стенки* из *event*(1) (соответственно *time*, *ball*, *wall*)
2. изменить положение объектов по значению *времени* (*time*)
3. определить следующее событие для *шарика* (*ball*)
4. подставить значения *шарика*, *времени* и *стенки* (соответственно *ball*, *time*, *wall*) в *event*
5. показать новое состояние системы на экране until нажатие клавиши

Этот цикл, состоящий из пяти шагов, повторяется до тех пор, пока не будет нажата соответствующая клавиша. О том, как пользоваться клавишами для прерывания программы, мы говорили в статье, посвященной специальным видеоэффектам и опубликованной в февральском номере журнала за этот год.

Шаг 1 довольно прост. На самом деле очередь событий состоит из трех массивов. В массиве *event* (событие) хранятся значения временных интервалов от данного момента до следующих событий в их естественном порядке. В массиве *ball* (шарики) хранятся индексы тех шариков, которые соответствуют событиям массива *event*. Еще один массив, под названием *wall* (стенка), содержит в качестве своих элементов индекс стенки, соответствующей событию. Например, элемент *event* (2) содержит время до второго события, *ball* (2) содержит номер участвующего в этом событии шарика b, а *wall* (2) указывает номер стенки w, который может принимать одно из четырех значений: 1 ($x = 0$), 2 ($x = 50$), 3 ($y = 0$) или 4 ($y = h$). Таким образом, шаг 1 реализуется тремя инструкциями, присваивающими переменным t, b и w три соответствующих элемента информации.

На шаге 2 нужно изменить положение всех шариков, содержащихся в цилиндре. Это можно сделать при помощи цикла, имеющего следующий вид:

for $i \leftarrow 1$ to n
$x(i) \leftarrow x(i) + t \cdot v_x(i)$
$y(i) \leftarrow y(i) + t \cdot v_y(i)$
if $w = 1$ or $w = 2$ then $v_x(b) \leftarrow -v_x(b)$
if $w = 3$ or $w = 4$ then $v_y(b) \leftarrow -v_y(b)$

Читатели, наверное, узнали рассмотренные выше формулы для вычисления положения шариков. Ни один шарик не сможет выйти за пределы сосуда, поскольку за время t он может лишь достигнуть стены, но не более. Два условных оператора if-then прямо обрабатывают отскок шарика b от

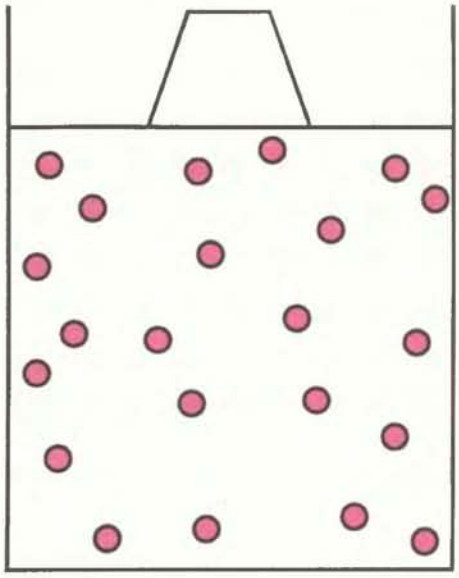


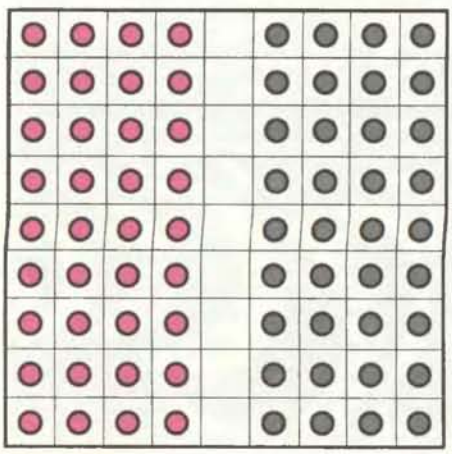
Схема цилиндра с поршнем

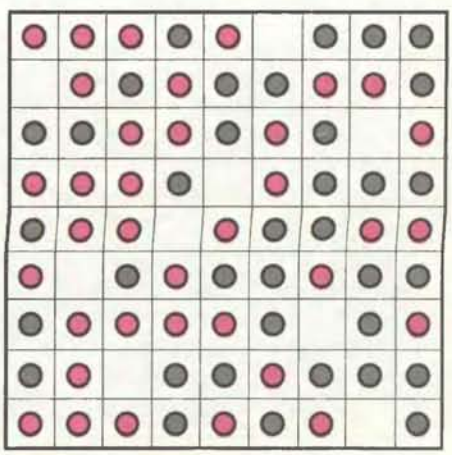
стенки. Если шарик ударяется об одну из вертикальных стенок, то составляющая его скорости по оси x меняется на противоположное значение, соударение же с горизонтальной стенкой означает, что поменять на противоположную следует координату скорости по оси y. При вычислении нового расположения объектов системы нужно также уменьшить величины времени в массиве событий *event*. Программа BOUNCE просто вычитает значение t из каждого элемента массива.

Шаг 3, определение следующего события для шарика b, я уже описал неявным образом; теперь рассмотрим его более подробно:

if $v_x(b)=0$ then $t_1 \leftarrow 5000$
$t_2 \leftarrow 5000$
else $t_1 \leftarrow x(b)/v_x(b)$
$t_2 \leftarrow (50 - x(b))/v_x(b)$
if $v_y(b)=0$ then $t_3 \leftarrow 5000$
$t_4 \leftarrow 5000$
else $t_3 \leftarrow -y(b)/v_y(b)$
$t_4 \leftarrow (h - y(b))/v_y(b)$

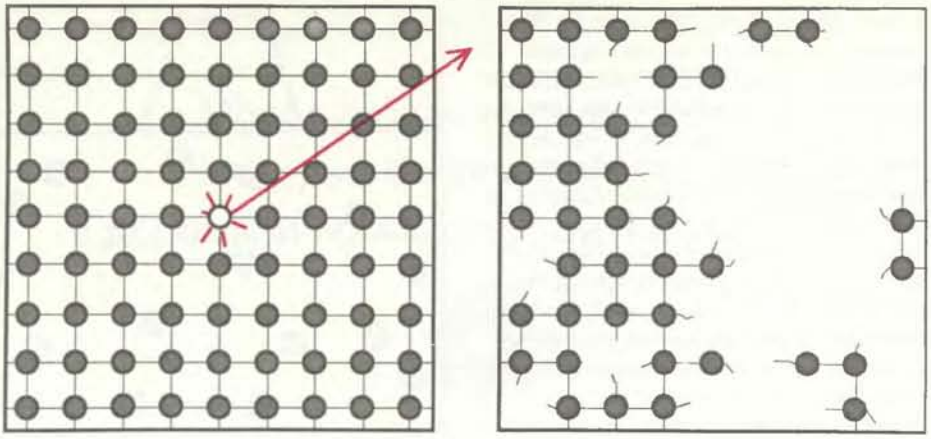
Время до столкновения между шари-





Процесс диффузии двух жидкостей в матрице 9 × 9

WWW.SCIAM-MAGAZINE.NAROD.RU



Фрагмент кристаллической решетки гридиума до (слева) *и после* (справа) *цепной реакции*

ком b и каждой из четырех стенок записано в виде значений четырех переменных от t_1 до t_4. Если шарик b на продолжении своего теперешнего пути никогда не достигнет какой-нибудь из четырех стенок, то соответствующей переменной присваивается абсурдно большое значение 5000. Затем, чтобы определить стенку, которой шарик b достигнет первой, мы применяем своеобразное решето. Сначала программа присваивает переменной t значение 5000 и сравнивает его по очереди с каждым из четырех параметров от t_1 до t_4. Проверка состоит из трех условий: если t_i меньше t, если t_i больше нуля и если $wall(b)$ не равно i, то программа заменяет текущее значение t на t_i, а вместо w подставляет значение i.

На шаге 4 происходит перетасовка очередей. Главное — определить то место в очереди, которое соответствует следующему столкновению шарика b. Это сделать нетрудно:

```
k ← 2
while event (k) < t
  event (k − 1) ← event (k)
  ball (k − 1) ← ball (k)
  wall (k − 1) ← wall (k)
  k ← k + 1
  event (k − 1) ← t
  ball (k − 1) ← b
  wall (k − 1) ← w
```

Здесь программа просматривает очередь событий *event*, чтобы найти место, принадлежащее в данный момент шарику b. В процессе поиска она сдвигает все три массива на одну ячейку влево. Если программа находит элемент k, такой, что *event* (k) не меньше t, то очевидно, что время до следующего столкновения шарика b принадлежит предшествующей ячейке k. (Когда мы задаем начальное положение и скорости шариков, находящихся в цилиндре, нужно установить также исходные значения элементов очереди. Чтобы не делать программу слишком сложной, исходные положения и скорости можно задать вручную, а временные интервалы для ближайших событий вычислить, а затем ввести их в программу с клавиатуры.)

На последнем шаге цикла программа BOUNCE должна показать на экране цилиндр и все шарики в положениях, занимаемых ими на данный момент. Выполняя эту простую операцию, нужно, конечно, не забыть стереть изображения шариков в их старых положениях.

Перед тем как приступить к выполнению основного цикла вычислений, программа просит пользователя ввести значение скорости, с которой будут двигаться шарики. Если обозначать эту величину через v, то программа затем выбирает n случайных чисел в диапазоне от $-v$ до v и присваивает эти числа в качестве значений координат скоростей по оси x, т. е. $v_x(i)$. Составляющие скоростей по оси y вычисляются теперь по теореме Пифагора, а именно они полагаются равными квадратному корню из разности $v^2 - v_x^2(i)$. Возможно, читатели захотят поэкспериментировать с различными значениями v в диапазоне от 1 до 10.

Разумеется, авторы программы BOUNCE не устоят перед соблазном понаблюдать за тем, как поршень перемещается то вверх, то вниз в ответ на давление, оказываемое на него снизу шариками, прыгающими в цилиндре. Для этого на шаге 2 нужно провести вычисления, в которых учитывается взаимодействие двух движений, совершаемых поршнем: вниз под влиянием силы тяжести и вверх под влиянием импульса, полученного при столкновении с очередным шариком. В результате мы получаем новое значение высоты h.

В целях упрощения программы физический процесс столкновений шариков с поршнем был сведен к примитивным арифметическим действиям. За время t поршень падает на $t^2/100$ единиц, а когда по нему ударяет шарик с вертикальной составляющей скорости v_y, он поднимается на высоту, равную $v_y/100\ m$ единиц. Я приношу свои извинения физикам, но мой кустарный способ работает! Разумеется, пользователи программы BOUNCE должны задать массу m поршня, а также количество шариков n и их скорость v.

В программе, моделирующей диффузию, под названием BLEND (смесь) молекулы жидкости рассматриваются как шарики, которые могут перемещаться вертикально или горизонтально лишь на одну позицию за раз. Массив размером 31 × 31 под названием *tank* (бак) хранит данные о 930 шариках, исходное расположение которых напоминает расстановку шашек перед началом игры. Программа BLEND еще менее реалистична, чем BOUNCE, но она тоже работает. Шарики, находящиеся в левой половине замкнутого сосуда, окрашены в красный цвет, а те, что справа — в серый (см. нижний рисунок на с. 87).

Начав работать, программа BLEND выбирает случайным образом позицию в массиве шариков, а затем также случайно одну из четырех соседних позиций. Если в первой позиции оказывается какой-нибудь шарик, а вторая пуста, то изображение шарика в старой позиции стирается и он появляется в новой, соседней позиции. Если шарик отсутствует в первой позиции и присутствует во второй, программа оставляет эти позиции без изменений. Через несколько микросекунд найдется какой-нибудь шарик, который она сможет переместить на соседнее место. Сколько времени потребуется для того, чтобы эти две абстрактные жидкости полностью перемешались? Мы не будем приводить здесь серьезного алгоритмического описания программы. Предоставляем читателям возможность проявить свои способности.

Мне не терпится поскорее перейти к

СТЕНКА	ВРЕМЯ t
1: $x = 0$	t_1: $-x/v_x$
2: $x = 50$	t_2: $(50 - x)/v_x$
3: $y = 0$	t_3: $-y/v_y$
4: $y = h$	t_4: $(h - y)/v_y$

СТЕНКА	ТОЧКА СТОЛКНОВЕНИЯ
1: $x = 0$	$(0, y + t_1 \cdot v_y)$
2: $x = 50$	$(50, y + t_2 \cdot v_y)$
3: $y = 0$	$(x + t_3 \cdot v_x, 0)$
4: $y = h$	$(x + t_4 \cdot v_x, h)$

Таблицы для расчета времени (слева) *и места* (справа) *столкновений*

WWW.SCIAM-MAGAZINE.NAROD.RU

гридиуму, очень опасному, нестабильному физического элементу. При спонтанном распаде одного атома этого вещества выделяется один нейтрон. Здесь я не могу не вспомнить о знаменитом опыте Уолта Диснея. Однажды этот великий затейник заполнил весь школьный спортивный зал взведенными мышеловками. На захлопывающейся рамке каждой мышеловки покоился шарик для игры в настольный теннис. С озорной улыбкой Дисней бросил в зал теннисный шарик. Через секунду в воздухе оказались тысячи шариков и зал наполнился каким-то странным шумом. Наверное, это была самая безопасная в мире цепная реакция.

Гридиум имеет кристаллическую структуру. Его атомы, подобные маленьким шарикам, расположены в виде квадратной решетки со стороной $n + 1$ шариков. По сигналу пользователя программы центральный шарик распадается и из него вылетает нейтрон, который стремительно движется сквозь решетку шариков в случайном направлении. Если он натолкнется по пути на другой шарик, тот тоже подвергнется распаду. Начнется ли цепная реакция? Если решетка мала, то реакции скорее всего не получится. Но чем больше шариков в решетке, тем с большей вероятностью она «взорвется» (см.рисунок на с. 88). Каковы же пороговые размеры, необходимые для начала цепной реакции? Предлагаем любителям экспериментов самим получить ответ на этот вопрос. Заметим, что даже маленький микрокомпьютер значительно облегчил бы работу физикам, исследовавшим цепную реакцию в 1944 г. в Лос-Аламосе.

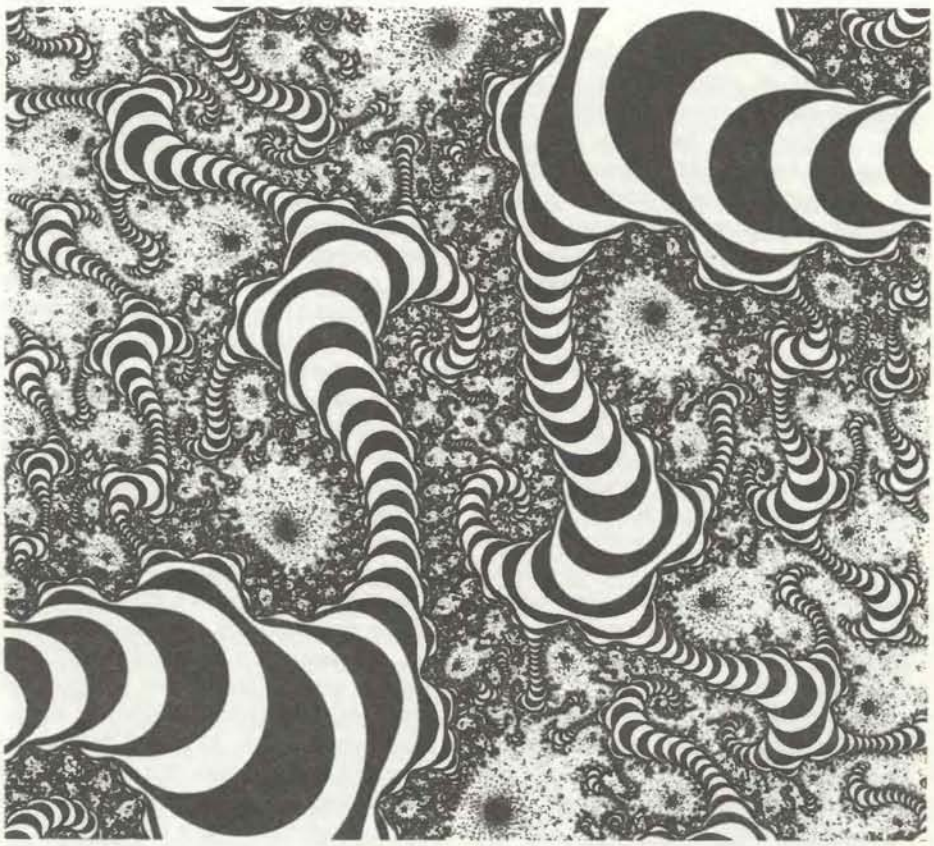
В нашей рубрике мы ничего не скрываем от наших читателей, у нас нет военных тайн. Я с удовольствием опубликую результаты исследований наших читателей, которым удастся определить критическую массу гридиума. В программе, которую я назвал BOOM (взрыв), используется массив *grid* (решетка). Поскольку в языках программирования, которыми пользуется большинство читателей, отсутствуют массивы переменной размерности, я решил установить для массива *grid* максимальные размеры 50 × 50. Когда пользователь введет с клавиатуры значение n, программа автоматически ограничит свои операции первыми n строками и столбцами массива *grid*. В каждом элементе массива содержится по одному атому. Чтобы упростить вычисления и демонстрацию картинки на экране, радиус атома гридиума примем равным единице, а интервал между соседними атомами — шести единицам. Для удобства будем также считать, что атом, содержащийся в элементе (i, j) массива, имеет координаты $(6i, 6j)$.

Когда атом гридиума претерпевает радиоактивный распад, программа BOOM выбирает случайным образом угол α и вычисляет синус и косинус этого угла, записывая их в качестве значений переменных s и c. При известных координатах атома (x, y) и угла нетрудно построить прямую линию. Важное значение для нас будет иметь вопрос о том, «насколько близко эта прямая пройдет от другого атома, имеющего координаты (u, v)?» Ответ дает следующая формула: $(u - x)^2 + (v - y)^2 - [(u - x) \cdot c + (v - y) \cdot s]^2$.

По этой формуле вычисляется квадрат расстояния между точкой (u, v) и рассматриваемой нами прямой. Вместо того чтобы возиться с извлечением квадратного корня, программа просто проверяет, не превосходит ли значение этой величины единицу. Если нет, то ясно, что атом с координатами (u, v) также должен претерпеть распад.

Всякий раз, когда один из атомов решетки *grid* взрывается, программа BOOM входит в цикл, в котором в переменные u и v систематически подставляются координаты всех n^2 атомов, за исключением взорвавшегося. Атомы, которые должны претерпеть распад, добавляются к списку, имеющему форму массива и имя *fire* (огонь). Индекс *fired* (сгорели) отделяет уже взорвавшиеся атомы, значащиеся в этом списке, от еще не взорвавшихся. Еще один индекс *end* указывает на конец списка. Каждый раз, когда взрывается новый атом, все другие атомы, лежащие на пути испущенного им нейтрона, добавляются к списку *fire*, при этом значение индекса *end* соответственно увеличивается. Когда все такие атомы определены, увеличивается значение индекса *fired* и описанный цикл повторяется вновь. Если значения индексов *end* и *fired* равны, они оба указывают количество атомов, принявших участие в цепной реакции. При каком значении n вся решетка полностью прореагирует? По-видимому, статистический разброс результатов вокруг правильного ответа будет небольшим.

Ввиду того что мы уже почти исчерпали отведенное для нашей статьи место, на этом описание программы BOOM придется завершить. Можно, конечно, вставить в программу процедуру вывода изображения на экран, но для того чтобы найти ответ на поставленный вопрос, это не обязательно. В одном из следующих номеров я опубликую наиболее интересное изображение частично взорванной решетки из числа тех, которые пришлют читатели. Я думаю, что значение n для этой решетки окажется очень близким к критической массе.



Изображение множества Мандельброта, полученное Дж. Лойлессом

WWW.SCIAM-MAGAZINE.NAROD.RU

В СТАТЬЕ январского номера мы вновь обратились к множеству Мандельброта и рассказали о мириадах родственных ему множеств, носящих имя Жюлиа, французского математика, одного из ведущих специалистов в области динамических систем. Программы, порождающие множества Жюлиа, настолько же просты, как и те, с помощью которых можно получать множества Мандельброта. Поэтому сотни наших читателей, а может быть уже и тысячи, написали свои собственные программы или собираются сделать это. В США издается специальный журнал о фракталах под названием "AMYGDALA". В нем приводятся различные полезные советы для тех, кто пишет программы, которые порождают множество Мандельброта или другие фрактальные множества. На его страницах программисты делятся своими секретами и рекламируют написанные ими программы.

В конце упомянутой статьи я продемонстрировал трехмерное чудовище, которое в действительности представляло собой поперечное сечение четырехмерного множества Мандельброта. Хайнц-Отто Пайтген, западногерманский математик и художник, раскрыл «рецепт» четырехмерного множества. Нужно итерировать уравнение $z \leftarrow z^3 - 3a^2z + b$, где a и b комплексные параметры. Какие точки играют в четырехмерном множестве ключевую роль подобно точке $z = 0$? На самом деле существуют сразу две такие точки, а именно $z = +a$ и $z = -a$. Следовательно, мы имеем не одно, а два четырехмерных множества Мандельброта.

Г. Тичаут из Эверетта (шт. Вашингтон) пишет, что он рассматривает множества Мандельброта и Жюлиа как сечения четырехмерного множества, хотя его четырехмерное множество не совпадает с множеством Пайтгена. Основная итерационная формула $z \leftarrow z^2 + c$ порождает множество Мандельброта, когда z остается постоянным, а c варьирует. Она же порождает множество Жюлиа, если c фиксировано, а z варьирует. Но z и c вместе определяют четыре измерения, по два на каждый параметр. Если собрать вместе все множества Жюлиа не в одномерную, а в двумерную совокупность, то возникает супермножество. М. Хармен из Кемберли (Англия) поделился размышлениями о таком же объекте.

Однако четырехмерный приз следует все же присудить У. Дженксу из Стамфорда (шт. Коннектикут). Не удовлетворенный обычными комплексными числами, Дженкс изобрел свои собственные числа, состоящие из четырех частей. Его числа аналогичны так называемым кватернионам, сложному множеству чисел, которое было изобретено математиками прошлого столетия и которым теперь пользуются физики. Дженкс прислал мне трехмерные представления своего объекта в виде последовательности из 29 его сечений.

Конечно, читатели, увлекающиеся построением изображений множества Мандельброта, хотели бы терять при этом поменьше времени — ведь кроме изображений итерационный процесс породил и множество обкусанных от нетерпения ногтей. Д. Холт из Силвер-Спринга (шт. Мэриленд) обнаружил, что скорость можно повысить на 8%, если вместо процедуры возведения числа в квадрат воспользоваться обычной операцией умножения числа само на себя. В его программном обеспечении эта операция выполняется быстрее, чем возведение в квадрат.

Наконец, рисунок внизу на с. 89 представляет собой еще один экспонат в постоянно пополняющейся коллекции изображений множества Мандельброта. Он прислан Дж. Лойлессом из Лилберна (шт. Джорджия). При этом он воспользовался системой «XT турбо-клоун с арифметическим сопроцессором 8087 и 24-точечным принтером Тошиба 321».

Наука и общество

НАСА в плену проблем

ОДИН астрофизик, сотрудник НАСА, касаясь наблюдений сверхновой 1987A, ведущихся с аэростатов, самолетов и спутников, заметил: «Это единственная область, где, судя по всему, у нас что-то получается».

В самом деле, спустя два года после взрыва «Челленджера» несчастливая звезда НАСА «светит» как никогда. Планы этой организации на ближайшие годы отягощены рядом технических проблем, с которыми она столкнулась при попытках разработать новый шаттл с двигателями на твердом топливе. Как заявили официальные представители НАСА, из-за этих трудностей запуски шаттлов, которые должны были возобновиться в июне этого года, задержатся по меньшей мере на два месяца. Другие специалисты считают, что задержка окажется намного большей. Слабое руководство со стороны Белого дома создает еще более серьезную и далеко идущую проблему. Ожидалось, что президент Рейган в своем обращении к народу 25 января даст широкую панораму перспектив США в области освоения космического пространства, но он остановился только на вопросах, связанных со стратегической оборонной инициативой. На следующий день предполагалось провести пресс-конференцию, на которой руководители НАСА и министерств торговли и транспорта должны были ознакомить общественность с новыми планами администрации по исследованию космоса, но Белый дом отменил это мероприятие за час до его начала.

В интервью корреспонденту журнала "Scientific American" У. Грехем, советник президента по науке, вкратце изложил основные концепции новой политики администрации в этой области. Он отметил, что появившиеся в прессе прогнозы, будто узловыми аспектами этой политики будет «заселение» Луны и пилотируемый полет к Марсу, возможно, совместно с СССР, далеки от истины. Грехем заявил, что администрация умышленно избежала выбора какой-либо конкретной цели, на достижение которой НАСА могло бы направить свои усилия. «Несмотря на свою привлекательность, — указал он, — программа по высадке людей на Луну оказалась несостоятельной в качестве задачи на далекую перспективу».

Кроме того, НАСА запрещено вести разработки систем и особенно средств доставки, которые оно может купить у частных фирм. «Мы не хотим, чтобы НАСА конкурировало с частным предпринимательством», — сказал Грехем. Политика администрации, похоже, основывается на допущении, что налагаемые на НАСА ограничения будут содействовать тому, чтобы более эффективный частный сектор вкладывал бóльшие средства в освоение космического пространства. Эта концепция «приватизации», по мнению ряда специалистов, отражает возрастающее влияние министерства торговли, которое с 1984 г. уполномочено конгрессом содействовать всем коммерческим предприятиям в области совершенствования космической техники. Как